EXHIBIT 14

US009270790B2

(12) **United States Patent**
Balassanian

(10) **Patent No.:** **US 9,270,790 B2**
(45) **Date of Patent:** **Feb. 23, 2016**

(54) METHOD AND SYSTEM FOR DATA DEMULTIPLEXING

(71) Applicant: **IMPLICIT, LLC**, Seattle, WA (US)

(72) Inventor: **Edward Balassanian**, Seattle, WA (US)

(73) Assignee: **Implicit, LLC**, Seattle, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 49 days.

(21) Appl. No.: **14/230,952**

(22) Filed: **Mar. 31, 2014**

(65) **Prior Publication Data**

US 2015/0009997 A1 Jan. 8, 2015

Related U.S. Application Data

(63) Continuation of application No. 13/911,324, filed on Jun. 6, 2013, now Pat. No. 8,694,683, which is a continuation of application No. 13/236,090, filed on Sep. 19, 2011, now abandoned, which is a continuation of application No. 10/636,314, filed on Aug. 6, 2003, now Pat. No. 8,055,786, which is a continuation of application No. 09/474,664, filed on Dec. 29, 1999, now Pat. No. 6,629,163.

(51) **Int. Cl.**

H04L 29/06	(2006.01)
H04L 12/701	(2013.01)
H04L 29/08	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 69/08*** (2013.01); ***H04L 29/06*** (2013.01); ***H04L 45/00*** (2013.01); ***H04L 69/22*** (2013.01); ***H04L 69/32*** (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,298,674	A	3/1994	Yun
5,414,833	A	5/1995	Hershey et al.
5,627,997	A	5/1997	Pearson et al.
5,761,651	A	6/1998	Hasebe
5,826,027	A	10/1998	Pedersen et al.
5,835,726	A	11/1998	Shwed et al.
5,848,233	A	12/1998	Radia et al.
5,848,415	A	12/1998	Guck
5,854,899	A	12/1998	Callon et al.
5,898,830	A	4/1999	Wesinger, Jr. et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0807347	11/1997
EP	0817031	1/1998

OTHER PUBLICATIONS

Alexander, D. et al., "The SwitchWare Active Network Architecture", Jun. 6, 1998, IEEE.

(Continued)

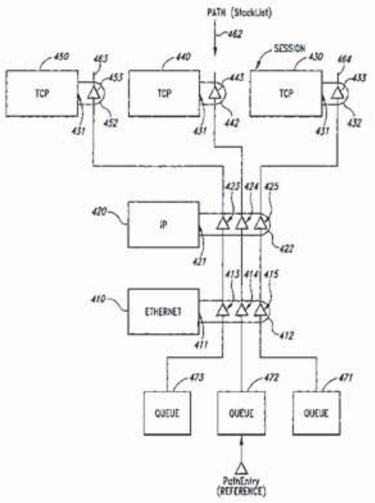
Primary Examiner — Duc Duong

(74) *Attorney, Agent, or Firm* — Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.

(57) ABSTRACT

A method and system for demultiplexing packets of a message is provided. The demultiplexing system receives packets of a message, identifies a sequence of message handlers for processing the message, identifies state information associated with the message for each message handler, and invokes the message handlers passing the message and the associated state information. The system identifies the message handlers based on the initial data type of the message and a target data type. The identified message handlers effect the conversion of the data to the target data type through various intermediate data types.

20 Claims, 16 Drawing Sheets



path through the bindings in the binding list. The routine loops selecting each path through the binding. The selected path is eligible if it starts at the first binding in the binding list and the path ends at the binding. The routine loops setting the short entry to the shortest eligible path found so far. In block **1505**, the routine sets the variable first binding to the first binding in the binding list of the path address. In block **1506**, the routine selects the next path (entry) in the path list of the binding starting with the first. If a path is selected (indicating that there are more paths in the binding), then the routine continues at block **1507**, else the routine continues at block **1509**. In block **1507**, the routine determines whether the selected path starts at the first binding in the binding list, whether the selected path ends at the last binding in the binding list, and whether the number of path entries in the selected path is less than the number of path entries in the shortest path selected so far. If these conditions are all satisfied, then the routine continues at block **1508**, else the routine loops to block **1506** to select the next path (entry). In block **1508**, the routine sets the shortest path (short entry) to the selected path and loops to block **1506** to select the next path through the binding. In block **1509**, the routine sets the selected path (entry) to the shortest path. In decision block **1510**, if a path has been found, then the routine continues at block **1511**, else the routine continues at block **1512**. In block **1511**, the routine sets the path to the path of the selected path entry and returns. Blocks **1512-1516** are performed when no paths have been found. In block **1512**, the routine sets the path to the path of the passed path entry. If the passed path entry has a path and its status is demux extend, then the routine continues at block **1515**, else the routine continues at block **1513**. In block **1513**, the routine creates a path for the path address. In block **1514**, the routine sets the variable element to null and sets the path entry to the first element in the stack list of the path. In block **1515**, the routine sets the variable element to be address entry of the member of the passed path entry and sets the path entry to the passed path entry. In block **1516**, the routine invokes the extend path routine to extend the path and then returns. The extend path routine creates a path through the bindings of the binding list and sets the path status to the current demux status.

FIG. **16** is a flow diagram of the process of path hopping routine. Path hopping occurs when the path through the binding list is not the same path as that of the passed path entry. In decision block **1601**, if the path of the passed path entry is set, then the routine continues at block **1602**, else the routine continues at block **1609**. In decision block **1602**, if the path of the passed path entry is equal to the local path, then the routine continues at **1612**, else path hopping is occurring and the routine continues at block **1603**. In blocks **1603-1607**, the routine loops positioning pointers at the first path entries of the paths that are not at the same binding. In block **1603**, the routine sets the variable old stack to the stack list of the path of the passed path entry. In block **1604**, the routine sets the variable new stack to the stack list of the local path. In block **1605**, the routine sets the variable old element to the next element in the old stack. In block **1606**, the routine sets the variable element to the next element in the new stack. In decision block **1607**, the routine loops until the path entry that is not in the same binding is located. In decision block **1608**, if the variable old entry is set, then the routine is not at the end of the hopped from path and the routine continues at block **1609**, else routine continues at block **1612**. In block **1609**, the routine sets the variable entry to the previous entry in the hopped-to path. In block **1610**, the routine sets the path of the passed path entry to the local path. In block **1611**, the routine sets the local entry to the first path entry of the stack list of the local path. In block **1612**, the routine inserts an entry into return list and then returns.

Although the conversion system has been described in terms of various embodiments, the invention is not limited to these embodiments. Modification within the spirit of the invention will be apparent to those skilled in the art. For example, a conversion routine may be used for routing a message and may perform no conversion of the message. Also, a reference to a single copy of the message can be passed to each conversion routine or demuxkey routine. These routines can advance the reference past the header information for the protocol so that the reference is positioned at the next header. After the demux process, the reference can be reset to point to the first header for processing by the conversion routines in sequence. The scope of the invention is defined by the claims that follow.

What is claimed is:

1. An apparatus, comprising:

a processing unit; and

a memory storing instructions executable by the processing unit to:

identify a path for one or more received packets of a message, wherein the path indicates a sequence of two or more routines for processing packets in the message, wherein the path is identified based on a key located in one of the received packets, and wherein the key includes an IP address and a port address; and

process the one or more received packets using the sequence of routines indicated in the identified path, wherein the sequence includes a routine that is used to execute a Transmission Control Protocol (TCP) to convert one or more packets having a TCP format into a different format.

2. The apparatus of claim **1**, wherein the key includes a remote port address and a local port address.

3. The apparatus of claim **1**, wherein the sequence of routines includes:

a second routine that is used to execute a second, different protocol to convert packets of the different format into another format, wherein the second protocol is an application layer protocol.

4. The apparatus of claim **3**, wherein the sequence of routines further includes a third routine that is used to execute a different application layer protocol to further convert the packets.

5. The apparatus of claim **1**, wherein the path further indicates sessions corresponding to respective ones of the sequence of routines.

6. The apparatus of claim **1**, wherein the key identifies a TCP session associated with the received one or more packets.

7. The apparatus of claim **1**, wherein the sequence of routines includes a routine that is executable to process the one or more packets without converting a format of the packets.

8. An apparatus, comprising:

a processing unit; and

a memory storing instructions executable by the processing unit to:

receive one or more packets of a message;

identify, using an IP address and one or more port addresses located in one of the received packets, a sequence of two or more routines for processing packets in the message; and

process the one or more received packets using the identified sequence of routines, wherein the sequence includes a routine that is executable to perform a

Transmission Control Protocol (TCP) to convert at least one of the packets of the message into a different format.

9. The apparatus of claim **8**, wherein the one or more port addresses include a remote port address and a local port address.

10. The apparatus of claim **8**, wherein the sequence of routines includes a plurality of application-level routines.

11. The apparatus of claim **8**, wherein the IP address and the one or more port addresses located in one of the received packets forms a key value that identifies a TCP session associated with the one or more received packets.

12. The apparatus of claim **8**, wherein the instructions are executable to use the IP address and the one or more port addresses to identify sessions corresponding to various ones of the sequence of routines.

13. The apparatus of claim **8**, wherein the instructions are executable to use the IP address and the one or more port addresses to identify a corresponding queue for the message.

14. The apparatus of claim **8**, wherein the sequence of routines includes a routine that does not perform a format conversion on the one or more received packets.

15. A non-transitory, computer-readable medium comprising software instructions for processing a message, wherein the software instructions, when executed, cause a computer system to:

identify a path for one or more received packets of the message, wherein the path indicates a sequence of two or more routines for processing packets in the message,

wherein the path is identified based on a key value located in one of the received packets, and wherein the key value includes an IP address and one or more port addresses;

process the one or more received packets using the sequence of routines indicated in the identified path, wherein the sequence includes a routine that is used to execute a Transmission Control Protocol (TCP) to convert one or more packets having a TCP format into a different format.

16. The computer-readable medium of claim **15**, wherein the one or more port addresses in the key value include a remote port address and a local port address.

17. The computer-readable medium of claim **15**, wherein the path indicates sessions corresponding to respective ones of the sequence of routines.

18. The computer-readable medium of claim **15**, wherein the sequence of routines includes a plurality of application-level routines.

19. The computer-readable medium of claim **18**, wherein the plurality of application-level routines includes a decryption routine.

20. The computer-readable medium of claim **15**, wherein the sequence of routines includes a routine that is used to execute an Internet Protocol (IP) to convert packets having an IP format into the TCP format, and wherein the key value further identifies a TCP session associated with the one or more received packets.

* * * * *